PUBLIC
SQL Anywhere - MobiLink
Document Version: 17.01.0 – 2021-10-15

MobiLink - Java API Reference

THE BEST RUN SAP

Content

1 MobiLink Server Java API Reference

MobiLink server Java API topics explain interfaces and classes, and their associated methods and constructors. To use these classes, reference the `mlscript.jar` assembly, located in *%SQLANY17%*`\java\`.

Package

```
com.sap.ml.script
```

In this section:

UpdateResultSet Interface [page 68]
A result set object that includes special methods for accessing the pre-image (old) and post-image (new) values of a specified row.

UploadData Interface [page 71]
Encapsulates upload operations for direct row handling.

UploadedTableData Interface [page 74]
Encapsulates table operations for direct row handling uploads.

1.1 DBConnectionContext Interface

Obtains information about the current database connection.

Package

```
com.sap.ml.script
```

Syntax

```
public interface DBConnectionContext
```

Members

All members of DBConnectionContext, including inherited members.

Methods

Modifier and Type	Method	Description
public java.sql.Connection	getConnection() [page 8]	Returns the existing connection to the consolidated database.
public DownloadData	getDownloadData() [page 9]	Returns the DownloadData for the current synchronization.
public NetworkData	getNetworkData() [page 10]	Returns information about the network streams for a synchronization.
public Properties	getProperties() [page 10]	Returns the Properties for this connection based on its script version.
public String	getRemoteID() [page 11]	Returns the remote ID of the database currently synchronizing on this connection.

Modifier and Type	Method	Description
public ServerContext	getServerContext() [page 12]	Returns the ServerContext for this synchronization server.
public String	getVersion() [page 13]	Returns the version string for this connection.

Remarks

This is passed to the constructor of classes containing scripts. If context is required for a background thread or beyond the lifetime of a connection, use the *ServerContext* interface instead.

> **i** Note
>
> A *DBConnectionContext* instance should not be used outside the thread that calls into your Java code.

Example

The following example shows you how to create a class level *DBConnectionContext* instance to use in your synchronization scripts. The *DBConnectionContext.getConnection* method obtains a *DBConnection* instance representing the current connection with the consolidated database.

```
import com.sap.ml.script;
public class OrderProcessor {
   DBConnectionContext _cc;
   public OrderProcessor( DBConnectionContext cc ) {
       _cc = cc;
   }
   // The method used for the handle_DownloadData event.
   public void HandleEvent() {
       DBConnection my_connection = _cc.GetConnection();
       // ...
   }
   // ...
}
```

In this section:

getConnection() Method [page 8]
Returns the existing connection to the consolidated database.

getDownloadData() Method [page 9]
Returns the DownloadData for the current synchronization.

getNetworkData() Method [page 10]
Returns information about the network streams for a synchronization.

getProperties() Method [page 10]
Returns the Properties for this connection based on its script version.

getRemoteID() Method [page 11]

Returns the remote ID of the database currently synchronizing on this connection.

getServerContext() Method [page 12]
Returns the ServerContext for this MobiLink server.

getVersion() Method [page 13]
Returns the version string for this connection.

Related Information

ServerContext Interface [page 40]

1.1.1 getConnection() Method

Returns the existing connection to the consolidated database.

Syntax

```
public java.sql.Connection getConnection () throws SQLException
```

Returns

An existing connection as a JDBC connection.

Exceptions

java.sql.SQLException Thrown when an error occurred binding the existing connection as a JDBC connection.

Remarks

This connection is the same connection that the synchronization server uses when executing SQL scripts for this synchronization.

This connection must not by committed, closed or altered in any way that would affect the synchronization server use of this connection. The connection returned is only valid for the lifetime of the underlying connection.

> **Note**
>
> Do not use the connection after the end_connection event has been called for that connection.

Related Information

makeConnection() Method [page 49]
DBConnectionContext Interface [page 6]

1.1.2 getDownloadData() Method

Returns the DownloadData for the current synchronization.

> **Syntax**

```
public DownloadData getDownloadData ()
```

Returns

DownloadData for the current synchronization. Null if this synchronization has no download.

Remarks

Use the DownloadData class to create the download for direct row handling.

Example

The following example shows you how to obtain a *DownloadData* instance for the current synchronization using the *DBConnectionContext.getDownloadData* method. This example assumes you have created a *DBConnectionContext* instance named *_cc*.

```
// The method used for the handle_DownloadData event.
public void HandleDownload() throws SQLException {
   // Get the DownloadData for the current synchronization
   DownloadData my_dd = _cc.getDownloadData();
   // ...
}
// ...
```

Related Information

DownloadData Interface [page 13]

1.1.3 getNetworkData() Method

Returns information about the network streams for a synchronization.

Syntax

```
public NetworkData getNetworkData ()
```

Returns

Information about the network streams used for the request, or null if the collection has not been enabled.

Remarks

This method is useful when authenticating against another server in the enterprise that uses the client-side certificate and HTTP headers.

To enable a collection of network stream data, add collect_network_data=1 to your -x switches. This option adds additional per-sync memory overhead to store the data.

Related Information

NetworkData Interface [page 33]

1.1.4 getProperties() Method

Returns the Properties for this connection based on its script version.

Syntax

```
public Properties getProperties ()
```

Returns

The properties for this connection.

Remarks

Properties are stored in the *ml_property* table.

Consult your Java Software Development Kit documentation for more information about *java.util.Properties*.

Example

The following example shows you how to output the properties for a *DBConnectionContext*. This example assumes you have a *DBConnectionContext* instance named *_cc*.

```
// The method used to output the connection properties.
public void outputProperties() {
   // Output the properties for the current synchronization
   java.util.Properties properties = _cc.getProperties();
   System.out.println(properties.toString());
}
```

1.1.5 getRemoteID() Method

Returns the remote ID of the database currently synchronizing on this connection.

Syntax

```
public String getRemoteID ()
```

Returns

The remote id.

Example

The following example shows you how to output the remote ID for a *DBConnectionContext*. This example assumes you have a *DBConnectionContext* instance named *_cc*.

```
// The method used to output the remote ID.
public void outputRemoteID() {
   // output the Remote ID for the current synchronization
   String remoteID = _cc.getRemoteID();
   System.out.println(remoteID);
}
```

1.1.6 getServerContext() Method

Returns the ServerContext for this MobiLink server.

Syntax

```
public ServerContext getServerContext ()
```

Returns

The MobiLink server context.

Example

The following example shows you how get the *ServerContext* instance for a *DBConnectionContext*. This example assumes you have a *DBConnectionContext* instance named *_cc*.

```
// A method that uses a ServerContext instance to shut down the server
public void shutDownServer() {
   ServerContext context = _cc.getServerContext();
   context.shutdown();
}
```

Related Information

ServerContext Interface [page 40]

1.1.7 getVersion() Method

Returns the version string for this connection.

Syntax

```
public String getVersion ()
```

Returns

The script version.

Example

The following example shows you how to get the script version and base decisions on its value. This example assumes you have a *DBConnectionContext* instance named *_cc*.

```
// A method that uses the script version
public void handleEvent() {
   // ...
   String version = _cc.getVersion();
   if (version.equals("My Version 1")) {
       // ...
   } else if (version.equals("My Version 2")) {
       // ...
   }
}
// ...
```

1.2 DownloadData Interface

Encapsulates download data operations for direct row handling.

Package

```
com.sap.ml.script
```

Syntax

```
public interface DownloadData
```

Members

All members of DownloadData, including inherited members.

Methods

Modifier and Type	Method	Description
public DownloadTableData	getDownloadTableByName(String) [page 15]	Gets the named download table for this synchronization.
public DownloadTableData[]	getDownloadTables() [page 16]	Gets an array of all the tables for download in this synchronization.

Remarks

Use the *DBConnectionContext.getDownloadData* method to obtain a *DownloadData* instance.

Use the *getDownloadTables* and *getDownloadTableByName* methods to return *DownloadTableData* instances

This download data is available through *DBConnectionContext*. It is not valid to access the download data before the begin_synchronization event or after the end_download event. It is not valid to access *DownloadData* in an upload-only synchronization.

Example

The following example shows you how to obtain a *DownloadData* instance for the current synchronization using the *DBConnectionContext.getDownloadData* method:

```
DBConnectionContext _cc;
// Your class constructor.
public OrderProcessor(DBConnectionContext cc) {
   _cc = cc;
}
// The method used for the handle_DownloadData event.
public void handleDownload() throws SQLException {
   // Get the DownloadData for the current synchronization.
   DownloadData my_dd = _cc.getDownloadData();
   // ...
}
```

In this section:

Related Information

getDownloadData() Method [page 9]
DownloadTableData Interface [page 17]

1.2.1 getDownloadTableByName(String) Method

Gets the named download table for this synchronization.

Syntax

```
public DownloadTableData getDownloadTableByName (String table_name)
```

Parameters

table_name The name of the table for which you want the download data.

Returns

A *DownloadTableData* instance representing the specified table, or null if a table of the given name does not exist for the current synchronization.

Example

The following example uses the *getDownloadTableByName* method to return a *DownloadTableData* instance for the *remoteOrders* table. This example assumes you have created a *DBConnectionContext* instance named *_cc*.

```
// The method used for the handle_DownloadData event.
public void handleDownload() throws SQLException {
   // Get the DownloadData for the current synchronization.
   DownloadData my_dd = _cc.getDownloadData();
   // Get the DownloadTableData for the remoteOrders table.
   DownloadTableData my_download_table =
my_dd.getDownloadTableByName("remoteOrders");
   // ...
}
```

Related Information

DownloadData Interface [page 13]
DownloadTableData Interface [page 17]
DBConnectionContext Interface [page 6]

1.2.2 getDownloadTables() Method

Gets an array of all the tables for download in this synchronization.

Syntax

```
public DownloadTableData[] getDownloadTables ()
```

Returns

An array of *DownloadTableData* objects for the current synchronization. The order of tables in the array is the same as the upload order of the remote.

Remarks

The operations performed on this table are sent to the remote database.

Example

The following example uses the *DownloadData.getDownloadTables* method to obtain an array of *DownloadTableData* objects for the current synchronization. The example assumes you have a *DBConnectionContext* instance named *_cc*.

```
// The method used for the handle_DownloadData event.
public void handleDownload() throws SQLException {
   // Get the DownloadData for the current synchronization.
   DownloadData my_dd = _cc.getDownloadData();
   // Get an array of tables to set download operations.
   DownloadTableData[] download_tables = my_dd.getDownloadTables();
   // Get the first table in the DownloadTableData array.
   DownloadTableData my_download_table = download_tables[0];
   // ...
}
```

Related Information

DownloadData Interface [page 13]
DownloadTableData Interface [page 17]
DBConnectionContext Interface [page 6]

1.3 DownloadTableData Interface

Encapsulates information for one download table for a synchronization.

Package

```
com.sap.ml.script
```

Syntax

```
public interface DownloadTableData
```

Members

All members of DownloadTableData, including inherited members.

Methods

Modifier and Type	Method	Description
public java.sql.PreparedStatement	getDeletePreparedStatement() [page 20]	Returns a java.sql.PreparedStatement instance that allows the user to add delete operations to the download.
public java.sql.Timestamp	getLastDownloadTime() [page 21]	Returns last download time for this table.
public java.sql.ResultSetMetaData	getMetaData() [page 22]	Gets the metadata for the DownloadTableData instance.
public String	getName() [page 23]	Returns the table name for the DownloadTableData instance.
public java.sql.PreparedStatement	getUpsertPreparedStatement() [page 24]	Returns a java.sql.PreparedStatement instance which allows the user to add upsert (insert or update) operations to the download of a synchronization.

Remarks

Use this interface to set the data operations that are downloaded to the client.

You can use the *DownloadData* interface to obtain *DownloadTableData* instances for the current synchronization. You can use the *getUpsertPreparedStatement* and *getDeletePreparedStatement* methods to obtain Java prepared statements for insert and update, and delete operations, respectively.

You can execute delete statement with all primary keys set to null to have the remote client truncate the table.

The *java.sql.PreparedStatement.executeUpdate* method registers an operation for download. Consult your Java Software Development Kit documentation for more information about *java.sql.PreparedStatement*.

> **i** Note
>
> You must set all column values for insert and update prepared statements. For delete operations you set primary key values. You cannot have both the delete and upsert prepared statements open at the same time.

Example

This example assumes that you have a table named *remoteOrders* in the synchronization client databases that was created using the following SQL statement:

```
CREATE TABLE remoteOrders (
   pk INT NOT NULL,
   col1 VARCHAR(200),
   PRIMARY KEY (pk)
);
```

The following example uses the *DownloadData.getDownloadTableByName* method to return a *DownloadTableData* instance representing the *remoteOrders* table. This example assumes you have a *DBConnectionContext* instance called *_cc*.

```
// The method used for the handle_DownloadData event.
public void handleDownload() throws SQLException {
   // Get the DownloadData for the current synchronization.
   DownloadData my_dd = _cc.getDownloadData();
   // Get the DownloadTableData for the remoteOrders table.
   DownloadTableData td = my_dd.getDownloadTableByName("remoteOrders");
   // User defined-methods to set download operations.
   setDownloadInserts(td);
   setDownloadDeletes(td);
   // ...
}
```

In this example, the *SetDownloadInserts* method uses *GetUpsertCommand* to obtain a command for the rows you want to insert or update. The *IDbCommand* holds the parameters that you set to the values you want inserted on the remote database.

```
void setDownloadInserts(DownloadTableData td) {
   java.sql.PreparedStatement insert_ps = td.getUpsertPreparedStatement();
   // The following method calls are the same as the following SQL statement:
   // INSERT INTO remoteOrders(pk, col1) values(2300, "truck");
```

```
    insert_ps.setInt(1, 2300);
    insert_ps.setString(2, "truck");
    int update_result = insert_ps.executeUpdate();
    if (update_result == 0) {
        // Insert was filtered because it was uploaded
        // in the same synchronization.
    }
    else {
        // Insert was not filtered.
    }
}
```

The *setDownloadDeletes* method uses the *DownloadTableData.getDeletePreparedStatement* to obtain a prepared statement for rows you want to delete. The *java.sql.PreparedStatement.setInt* method sets the primary key values for rows you want to delete in the remote database and the *java.sql.PreparedStatement.executeUpdate* method registers the row values for download.

```
void setDownloadDeletes(DownloadTableData td) {
    java.sql.PreparedStatement delete_ps = td.getDeletePreparedStatement();
    // The following method calls are the same as the following SQL statement:
    // DELETE FROM remoteOrders  where pk=2300;
    delete_ps.setInt(1, 2300);
    delete_ps.executeUpdate();
}
```

In this section:

Related Information

DownloadData Interface [page 13]

1.3.1 getDeletePreparedStatement() Method

Returns a java.sql.PreparedStatement instance that allows the user to add delete operations to the download.

Syntax

```
public java.sql.PreparedStatement getDeletePreparedStatement () throws
java.sql.SQLException
```

Returns

A *java.sql.PreparedStatement* instance for adding delete operations to the download.

Exceptions

java.sql.SQLException Thrown if there is a problem retrieving the deleted *java.sql.PreparedStatement* instance.

Remarks

The prepared statement applies to the *DownloadTableData* instance and contains a parameter for each primary key column in the table.

To include a delete operation in the download, set all columns in your *java.sql.PreparedStatement* and then call the *java.sql.PreparedStatement.executeUpdate* method.

Set all the parameters to null to have the remote database truncate the table.

Note

You must set all primary key values for download delete operations, or set all primary key values to null for truncate operations.

Example

In the following example, the *setDownloadDeletes* method uses the *getDeletePreparedStatement* to obtain a prepared statement for rows you want to delete. The *java.sql.PreparedStatement.setInt* method sets the primary key values for rows you want to delete in the remote database and the *java.sql.PreparedStatement.executeUpdate* method sets the row values in the download.

```
void setDownloadDeletes(DownloadTableData td) {
```

```
    java.sql.PreparedStatement delete_ps = td.getDeletePreparedStatement();
    // This is the same as executing the following SQL statement:
    // DELETE FROM remoteOrders  where pk=2300;
    delete_ps.setInt(1, 2300);
    delete_ps.executeUpdate();
    delete_ps.close();
}
```

Related Information

DownloadTableData Interface [page 17]

1.3.2 getLastDownloadTime() Method

Returns last download time for this table.

Syntax

```
public java.sql.Timestamp getLastDownloadTime ()
```

Returns

The last download time for this download table.

Remarks

This is the same last download time passed to several of the per table download events.

The last download time is useful for generating the table download data for a particular synchronization.

Example

The following shows you how to populate a table in the download with inserts using the last download time. Note that this example assumes you have a *DBConnectionContext* instance called *_cc*.

```
// The method used for the handle_DownloadData event.
public void handleDownload() throws SQLException {
    // Get the DownloadData for the current synchronization.
    DownloadData my_dd = _cc.getDownloadData();
    // Get the DownloadTableData for the remoteOrders table.
    DownloadTableData td = my_dd.getDownloadTableByName("remoteOrders");
```

```
    // Get the inserts given a last download time.
    ResultSet inserts_rs  = makeInsertsFromTimestamp(td.getLastDownloadTime());
    // Fill the DownloadTableData using the inserts resultset.
    setDownloadInsertsFromRS(td, inserts_rs);
    inserts_rs.close();
    // ...
}
```

Related Information

DownloadTableData Interface [page 17]

1.3.3 getMetaData() Method

Gets the metadata for the DownloadTableData instance.

Syntax

```
public java.sql.ResultSetMetaData getMetaData ()
```

Returns

The metadata for the *DownloadTableData* instance.

Remarks

The metadata is a standard *java.sql.ResultSetMetaData* object.

If you want the metadata to contain column name information, specify in your client that column names should be sent with the upload.

Consult your Java Software Development Kit documentation for more information about *java.sql.ResultSetMetaData*.

Example

The following example shows you how get the number of columns used in the query for the *DownloadTableData* instance:

```
import java.sql.ResultSetMetaData;
```

```
// The method used to return the number of columns in a DownloadTableData
instance query
public int getNumColumns(DownloadTableData td) {
   ResultSetMetaData rsmd = td.getMetaData();
   return rsmd.getColumnCount();
}
```

Related Information

DownloadTableData Interface [page 17]

1.3.4 getName() Method

Returns the table name for the DownloadTableData instance.

Syntax

```
public String getName ()
```

Returns

The table name for the *DownloadTableData* instance.

Remarks

You can also access the table name using the *java.sql.ResultSetMetaData* instance returned by the *getMetaData* method.

Example

The following example shows you how to output the table name for the *DownloadTableData* instance. This example assumes you have a *DBConnectionContext* instance named *_cc*.

```
// The method used for the handle_DownloadData event
public void handleDownload() throws SQLException {
// Get the DownloadData for the current synchronization.
DownloadData my_dd = _cc.getDownloadData();
// Get the DownloadTableData for the remoteOrders table.
DownloadTableData td = my_dd.getDownloadTableByName("remoteOrders");
// Print the table name to standard output (remoteOrders)
```

```
System.out.println(td.getName());
// User defined-methods to set download operations.
setDownloadInserts(td);
setDownloadDeletes(td);
// ...
}
```

Related Information

DownloadTableData Interface [page 17]
getMetaData() Method [page 22]

1.3.5 getUpsertPreparedStatement() Method

Returns a java.sql.PreparedStatement instance which allows the user to add upsert (insert or update) operations to the download of a synchronization.

Syntax

```
public java.sql.PreparedStatement getUpsertPreparedStatement () throws
java.sql.SQLException
```

Returns

A *java.sql.PreparedStatement* instance for adding upsert operations to the download.

Exceptions

java.sql.SQLException Thrown if there is a problem retrieving the upserted *java.sql.PreparedStatement* instance.

Remarks

The prepared statement applies to the DownloadTableData instance and contains a parameter for each column in the table.

To include an insert or update operation in the download, set all column values in your *java.sql.PreparedStatement* and then call the *java.sql.PreparedStatement.executeUpdate* method. Calling *java.sql.PreparedStatement.executeUpdate* on the prepared statement returns 0 if the insert or update

operation was filtered and returns 1 if the operation was not filtered. An operation is filtered if it was uploaded in the same synchronization.

> **i** Note
>
> You must set all column values for download insert and update operations.

Example

In the following example, the *setDownloadInserts* method uses the *getUpsertPreparedStatement* to obtain a prepared statement for rows you want to insert or update. The *java.sql.PreparedStatement.setInt* and *PreparedStatement.setString* methods set the column values, and the *PreparedStatement.executeUpdate* method sets the row values in the download.

```
void setDownloadInserts(DownloadTableData td) {
   java.sql.PreparedStatement insert_ps = td.getUpsertPreparedStatement();
   // This is the same as executing the following SQL statement:
   // INSERT INTO remoteOrders(pk, col1) VALUES (2300, "truck");
   insert_ps.setInt(1, 2300);
   insert_ps.setString(2, "truck");
   int update_result = insert_ps.executeUpdate();
   if (update_result == 0) {
       // Insert was filtered because it was uploaded
       // in the same synchronization.
   }
   else {
       // Insert was not filtered.
   }
   insert_ps.close();
}
```

Related Information

DownloadTableData Interface [page 17]

1.4 InOutInteger Interface

Passed into methods to enable the functionality of an in/out parameter passed to a SQL script.

Package

```
com.sap.ml.script
```

Syntax

```
public interface InOutInteger
```

Members

All members of InOutInteger, including inherited members.

Methods

Modifier and Type	Method	Description
public int	getValue() [page 27]	Returns the value of this integer parameter.
public void	setValue(int) [page 27]	Sets the value of this integer parameter.

Example

The following call to a synchronization system procedure registers a Java method named *handleError* as the script for the *handle_error connection* event when synchronizing the script version *ver1*:

```
CALL ml_add_java_connection_script(
   'ver1',
   'handle_error',
   'ExamplePackage.ExampleClass.handleError'
)
```

The following is the sample Java method named *handleError*. It processes an error based on the data that is passed in. It also determines the resulting error code.

```
public String handleError(
   ianywhere.ml.script.InOutInteger actionCode,
   int errorCode,
   String errorMessage,
   String user,
   String table)
{
int new_ac;
if (user == null) {
   new_ac = handleNonSyncError(errorCode, errorMessage);
} else if (table == null) {
   new_ac = handleConnectionError(errorCode, errorMessage, user);
}
else {
   new_ac = handleTableError(errorCode, errorMessage, user, table);
}
// Keep the most serious action code.
if (actionCode.getValue() < new_ac) {
   actionCode.setValue(new_ac);
}
}
```

In this section:

1.4.1 getValue() Method

Returns the value of this integer parameter.

Syntax

```
public int getValue ()
```

Returns

The value of this integer.

Related Information

InOutInteger Interface [page 25]

1.4.2 setValue(int) Method

Sets the value of this integer parameter.

Syntax

```
public void setValue (int new_value)
```

Parameters

new_value The value this integer should take.

Related Information

InOutInteger Interface [page 25]

1.5 InOutString Interface

Passed into methods to enable the functionality of an in/out parameter passed to a SQL script.

Package

```
com.sap.ml.script
```

Syntax

```
public interface InOutString
```

Members

All members of InOutString, including inherited members.

Methods

Modifier and Type	Method	Description
public String	getValue() [page 29]	Returns the value of this String parameter.
public void	setValue(String) [page 30]	Sets the value of this String parameter.

Example

The following call to a synchronization system procedure registers a Java method named *modifyUser* as the script for the *modify_user* connection event when synchronizing the script version *ver1*:

```
CALL ml_add_java_connection_script(
    'ver1',
    'modify_user',
    'ExamplePackage.ExampleClass.modifyUser'
)
```

The following is the sample Java method named *modifyUser*. It gets the user ID from the database and then uses it to set the user name.

```
public String modifyUser(InOutString io_user_name) throws SQLException {
    Statement uid_select = curConn.createStatement();
    ResultSet uid_result = uid_select.executeQuery(
        "SELECT rep_id FROM SalesRep WHERE name = '"
            + io_user_name.getValue() + "' "
    );
    uid_result.next();
    io_user_name.setValue(java.lang.Integer.toString(uid_result.getInt(1));
    uid_result.close();
    uid_select.close();
    return (null);
}
```

In this section:

1.5.1 getValue() Method

Returns the value of this String parameter.

Syntax

```
public String getValue ()
```

Returns

The value of this String parameter.

Related Information

InOutString Interface [page 28]

1.5.2 setValue(String) Method

Sets the value of this String parameter.

Syntax

```
public void setValue (String new_value)
```

Parameters

new_value The value for this String to take.

Related Information

InOutString Interface [page 28]

1.6 LogListener Interface

Used for catching messages that are printed to the log.

Package

```
com.sap.ml.script
```

Syntax

```
public interface LogListener
```

Members

All members of LogListener, including inherited members.

Methods

Modifier and Type	Method	Description
public void	messageLogged(ServerContext, LogMessage) [page 31]	Invoked when a message is printed to the log.

In this section:

1.6.1 messageLogged(ServerContext, LogMessage) Method

Invoked when a message is printed to the log.

Syntax

```
public void messageLogged (
    ServerContext sc,
    LogMessage message
)
```

Parameters

sc The context for the server that is printing the message.
message The *LogMessage* that has been sent to the synchronization server log.

1.7 LogMessage Class

Holds the data associated with a log message.

Package

```
com.sap.ml.script
```

Syntax

```
public class LogMessage
```

Members

All members of LogMessage, including inherited members.

Variables

Modifier and Type	Variable	Description
public static final int	ERROR	Indicates that the log message is an error.
public static final int	WARNING	Indicates that the log message is a warning.
public static final int	INFO	Indicates that the message log displays information.

Methods

Modifier and Type	Method	Description
public String	getText() [page 32]	Accesses the text associated with this message.
public int	getType() [page 33]	Accesses this message type.
public String	getUser() [page 33]	Accesses the user name associated with this message.

In this section:

1.7.1 getText() Method

Accesses the text associated with this message.

Syntax

```
public String getText ()
```

Returns

The main text of this message

1.7.2 getType() Method

Accesses this message type.

Syntax

```
public int getType ()
```

Returns

The type of this message, which can be either *LogMessage.ERROR*, *LogMessage.INFO*, or *LogMessage.WARNING*.

1.7.3 getUser() Method

Accesses the user name associated with this message.

Syntax

```
public String getUser ()
```

Returns

The user associated with this message. This value may be null if the message has no user.

1.8 NetworkData Interface

Contains information about the network streams for a synchronization.

Package

```
com.sap.ml.script
```

Syntax

```
public interface NetworkData
```

Members

All members of NetworkData, including inherited members.

Methods

Modifier and Type	Method	Description
public java.security.cert.CertPath	getCertificateChain() [page 36]	Returns a java.security.cert.CertPath object containing any certificates sent by the client.
public Map< String, List< String > >	getHTTPHeaders() [page 37]	Returns a Map object that maps header names to a List of header-values.
public String	getHTTPHeaderValue(String) [page 37]	Returns the value of the last header received by the server with the supplied name.
public List< String >	getHTTPHeaderValues(String) [page 38]	Returns all the header values received by the server associated with the supplied name.
public boolean	isEndToEndEncrypted() [page 38]	Determines if this synchronization is end-to-end encrypted.
public boolean	isHTTP() [page 39]	Determines if the synchronization uses HTTP or HTTPS.
public boolean	isTLS() [page 39]	Determines if this synchronization uses TLS.

Remarks

This interface is useful when authenticating against another server in the enterprise that uses the client-side certificate and HTTP headers.

To enable collection of network stream data, add collect_network_data=1 to your -x switches. This adds additional per sync memory overhead to store the data. If using TLS or HTTPS with client-side certificates, add trusted_certificates=<certificate file> to have the server ask the client to send a certificate during the TLS handshake, incurring a time and network cost.

You can obtain a NetworkData object by invoking the getNetworkData method of the DBConnectionContext class. When using HTTP or HTTPS, it contains the header data for the last HTTP request received by the server before the authenticate scripts are invoked.

Example

The following example illustrates how to get a NetworkData object from the DBConnectionContext object, and output the data.

```
public class OrderProcessor {
   DBConnectionContext _cc;
   public OrderProcessor( DBConnectionContext cc ) {
       _cc = cc;
   }
   // The method used for the authenticate_user event.
   public void AuthUser() {
       NetworkData nd = _cc.getNetworkData();
       if( nd != null ) {
           if( nd.isHTTP() ) {
               System.out.println( "http" );
               String user_agent = nd.getHTTPHeaderValue( "user-agent" );
               System.out.println( "  user-agent: " + user_agent.substring( 0,
user_agent.indexOf( '/' ) ) );
           } else {
               System.out.println( "no http" );
           }
           if( nd.isTLS() ) {
               System.out.println( "tls" );
               CertPath certs = nd.getCertificateChain();
               if( certs != null ) {
                   System.out.println( "  client-side cert:" );
                   int n = 1;
                   for( Certificate c : certs.getCertificates() ) {
                       System.out.println( "    cert " + n++ );
                       X509Certificate c509 = (X509Certificate) c;
                       System.out.println( "      Subject: " +
c509.getSubjectX500Principal().getName() );
                       System.out.println( "      Issuer: " +
c509.getIssuerX500Principal().getName() );
                   }
               } else {
                   System.out.println( "  no client cert" );
               }
           } else {
               System.out.println( "no tls" );
           }
           if( nd.isEndToEndEncrypted() ) {
               System.out.println( "e2ee" );
           } else {
               System.out.println( "no e2ee" );
           }
       } else {
           System.out.println( "NULL networkdata" );
       }
   }
}
```

In this section:

getHTTPHeaderValues(String) Method [page 38]
: Returns all the header values received by the server associated with the supplied name.

isEndToEndEncrypted() Method [page 38]
: Determines if this synchronization is end-to-end encrypted.

isHTTP() Method [page 39]
: Determines if the synchronization uses HTTP or HTTPS.

isTLS() Method [page 39]
: Determines if this synchronization uses TLS.

Related Information

DBConnectionContext Interface [page 6]

1.8.1 getCertificateChain() Method

Returns a java.security.cert.CertPath object containing any certificates sent by the client.

Syntax

```
public java.security.cert.CertPath getCertificateChain ()
```

Returns

A CertPath containing the X509 certificates that identify the client; returns null if no such certificates were provided.

Remarks

The certificates are all java.security.cert.X509Certificate objects.

This method returns a non-null value only if the isTLS method returns true, the client supplies a certificate using the "identity" stream parameter, and the trusted_certificates option is set on the server. A non-null CertPath value contains the certificates, ordered from the self-signed certificate to the peer certificate.

1.8.2 getHTTPHeaders() Method

Returns a Map object that maps header names to a List of header-values.

Syntax

```
public Map< String, List< String > > getHTTPHeaders ()
```

Returns

A Map containing all the headers received by the server.

Related Information

getHTTPHeaderValue(String) Method [page 37]
getHTTPHeaderValues(String) Method [page 38]

1.8.3 getHTTPHeaderValue(String) Method

Returns the value of the last header received by the server with the supplied name.

Syntax

```
public String getHTTPHeaderValue (String name)
```

Parameters

name The header name to return the value for.

Returns

The last header value associated with the supplied header name.

Related Information

getHTTPHeaderValues(String) Method [page 38]
getHTTPHeaders() Method [page 37]

1.8.4 getHTTPHeaderValues(String) Method

Returns all the header values received by the server associated with the supplied name.

Syntax

```
public List< String > getHTTPHeaderValues (String name)
```

Parameters

name The header name to return the values for.

Returns

The header values associated with the supplied header name.

Related Information

getHTTPHeaderValue(String) Method [page 37]
getHTTPHeaders() Method [page 37]

1.8.5 isEndToEndEncrypted() Method

Determines if this synchronization is end-to-end encrypted.

Syntax

```
public boolean isEndToEndEncrypted ()
```

Returns

True if this synchronization is end-to-end encrypted; otherwise, returns false.

1.8.6 isHTTP() Method

Determines if the synchronization uses HTTP or HTTPS.

Syntax

```
public boolean isHTTP ()
```

Returns

True if this synchronization uses HTTP or HTTPS; otherwise, returns false.

1.8.7 isTLS() Method

Determines if this synchronization uses TLS.

Syntax

```
public boolean isTLS ()
```

Returns

True if this synchronization uses TLS; otherwise, returns false.

1.9 ServerContext Interface

An instantiation of all the context that is present for the duration of the synchronization server.

Package

```
com.sap.ml.script
```

Syntax

```
public interface ServerContext
```

Members

All members of ServerContext, including inherited members.

Methods

Modifier and Type	Method	Description
public void	addErrorListener(LogListener) [page 42]	Adds the specified LogListener object to receive a notification when an error is printed.
public void	addInfoListener(LogListener) [page 43]	Adds the specified LogListener object from the list of Listener objects to receive a notification when info is printed.
public void	addShutdownListener(ShutdownListener) [page 44]	Adds the specified ShutdownListener object that is to receive notification before the server context is destroyed.
public void	addWarningListener(LogListener) [page 45]	Adds the specified LogListener object to receive a notification when a warning is printed.
public Properties	getProperties(String, String) [page 45]	Returns the set of properties associated with a given component and property set.
public Properties	getPropertiesByVersion(String) [page 46]	Returns the set of properties associated with the script version.
public Iterator	getPropertySetNames(String) [page 47]	Returns the list of property set names for a given component.
public Object[]	getStartClassInstances() [page 48]	Gets an array of the start classes that were constructed at server start time.

Modifier and Type	Method	Description
public java.sql.Connection	makeConnection() [page 49]	Opens and returns a new server connection.
public void	removeErrorListener(LogListener) [page 49]	Removes the specified LogListener object from the list of Listener objects that receive notifications when an error is printed.
public void	removeInfoListener(LogListener) [page 50]	Removes the specified LogListener object from the list of Listener objects that receive notifications when info is printed.
public void	removeShutdownListener(ShutdownListener) [page 51]	Removes the specified ShutdownListener object from the list of Listener objects that receive notifications before this ServerContext is destroyed.
public void	removeWarningListener(LogListener) [page 51]	Removes the specified LogListener object from the list of Listener objects that receive notifications when a warning is printed.
public void	shutdown() [page 52]	Forces the server to shut down.

Remarks

This context can be held as static data and used in a background thread. It is valid for the duration of the Java VM invoked by the synchronization server.

To access a *ServerContext* instance, use the *DBConnectionContext.getServerContext* method.

In this section:

addErrorListener(LogListener) Method [page 42]
Adds the specified LogListener object to receive a notification when an error is printed.

addInfoListener(LogListener) Method [page 43]
Adds the specified LogListener object from the list of Listener objects to receive a notification when info is printed.

addShutdownListener(ShutdownListener) Method [page 44]
Adds the specified ShutdownListener object that is to receive notification before the server context is destroyed.

addWarningListener(LogListener) Method [page 45]
Adds the specified LogListener object to receive a notification when a warning is printed.

getProperties(String, String) Method [page 45]
Returns the set of properties associated with a given component and property set.

getPropertiesByVersion(String) Method [page 46]
Returns the set of properties associated with the script version.

getPropertySetNames(String) Method [page 47]

Returns the list of property set names for a given component.

getStartClassInstances() Method [page 48]
Gets an array of the start classes that were constructed at server start time.

makeConnection() Method [page 49]
Opens and returns a new server connection.

removeErrorListener(LogListener) Method [page 49]
Removes the specified LogListener object from the list of Listener objects that receive notifications when an error is printed.

removeInfoListener(LogListener) Method [page 50]
Removes the specified LogListener object from the list of Listener objects that receive notifications when info is printed.

removeShutdownListener(ShutdownListener) Method [page 51]
Removes the specified ShutdownListener object from the list of Listener objects that receive notifications before this ServerContext is destroyed.

removeWarningListener(LogListener) Method [page 51]
Removes the specified LogListener object from the list of Listener objects that receive notifications when a warning is printed.

shutdown() Method [page 52]
Forces the server to shut down.

1.9.1 addErrorListener(LogListener) Method

Adds the specified LogListener object to receive a notification when an error is printed.

Syntax

```
public void addErrorListener (LogListener ll)
```

Parameters

ll The LogListener object to be notified on error.

Remarks

When an error is printed, the *LogListener.messageLogged(ServerContext, LogMessage)* method is called.

Related Information

messageLogged(ServerContext, LogMessage) Method [page 31]
LogMessage Class [page 31]

1.9.2 addInfoListener(LogListener) Method

Adds the specified LogListener object from the list of Listener objects to receive a notification when info is printed.

Syntax

```
public void addInfoListener (LogListener ll)
```

Parameters

ll The *LogListener* object to be notified on info.

Remarks

The *LogListener.messageLogged* method is called.

Example

The following code registers a *MyLogListener* object to receive notifications of info messages:

```
// ServerContext serv_context;
serv_context.addInfoListener(new MyLogListener(ll_out_file));
// The following code shows an example of processing those messages:
class MyLogListener implements LogListener {
   FileOutputStream _out_file;
   public TestLogListener(FileOutputStream out_file) {
       _out_file = out_file;
   }
   public void messageLogged(ServerContext sc, LogMessage msg) {
       String type;
       String user;
       try {
           if (msg.getType() == LogMessage.ERROR) {
               type = "ERROR";
           } else if (msg.getType() == LogMessage.WARNING) {
               type = "WARNING";
           } else if (msg.getType() == LogMessage.INFO) {
               type = "INFO";
```

```
            } else {
                type = "UNKNOWN!!!";
            }
            user = msg.getUser();
            if (user == null) {
                user = "NULL";
            }
            _out_file.write(("Caught msg type="
                + type
                + " user=" + user
                + " text=" +msg.getText()
                + "\n").getBytes()
            );
            _out_file.flush();
        }
        catch(Exception e) {
// if we print the exception from processing an info message,
// we may recurse indefinitely
if (msg.getType() != LogMessage.ERROR) {
   // Print some error output to the synchronization server log.
   e.printStackTrace();
}
        }
    }
}
```

Related Information

messageLogged(ServerContext, LogMessage) Method [page 31]

1.9.3 addShutdownListener(ShutdownListener) Method

Adds the specified ShutdownListener object that is to receive notification before the server context is destroyed.

Syntax

```
public void addShutdownListener (ShutdownListener sl)
```

Parameters

sl The ShutdownListener object to be notified on shutdown.

Remarks

On shutdown, the *ShutdownListener.shutdownPerformed(ServerContext)* method is called.

Related Information

ShutdownListener Interface [page 55]
shutdownPerformed(ServerContext) Method [page 56]

1.9.4 addWarningListener(LogListener) Method

Adds the specified LogListener object to receive a notification when a warning is printed.

Syntax

```
public void addWarningListener (LogListener ll)
```

Parameters

ll The LogListener object to be notified on warning.

Remarks

The *LogListener.messageLogged(ServerContext, LogMessage)* method is called.

Related Information

LogMessage Class [page 31]
messageLogged(ServerContext, LogMessage) Method [page 31]

1.9.5 getProperties(String, String) Method

Returns the set of properties associated with a given component and property set.

Syntax

```
public Properties getProperties (
    String component,
    String set
)
```

Parameters

component The component.
set The property set.

Returns

The set of properties, which may be empty.

Remarks

These properties are stored in the *ml_property* system table.

Example

The following example shows how to lists all the properties on a *ServerContext* instance.

```
import java.util.*;
ServerContext serverContext;
PrintStream out
Properties prop = serverContext.getProperties();
prop.list(out);
```

1.9.6 getPropertiesByVersion(String) Method

Returns the set of properties associated with the script version.

Syntax

```
public Properties getPropertiesByVersion (String script_version)
```

Parameters

script_version The script version for which to return the associated properties.

Returns

The set of properties associated with the given script version.

Remarks

These are stored in the *ml_property* system table. The script version is stored in the *property_set_name* column when the *component_name* is *ScriptVersion*.

Example

The following example shows how to lists all the properties on a *ServerContext* instance associated with a given script version:

```
import java.util.*;
ServerContext serverContext;
PrintStream out
Properties prop = serverContext.getPropertiesByVersion("MyScriptVersion");
prop.list(out);
```

1.9.7 getPropertySetNames(String) Method

Returns the list of property set names for a given component.

Syntax

```
public Iterator getPropertySetNames (String component)
```

Parameters

component The name of the component for which to list property names.

Returns

The list of property set names for the given component.

Remarks

These properties are stored in the *ml_property* system table.

Example

The following example shows how to lists all the properties on a *ServerContext* instance associated with a given component:

```
import java.util.*;
ServerContext serverContext;
PrintStream out
Properties prop = serverContext.getPropertySetNames("Component Name");
prop.list(out);
```

1.9.8 getStartClassInstances() Method

Gets an array of the start classes that were constructed at server start time.

Syntax

```
public Object[] getStartClassInstances ()
```

Returns

An array of start classes that were constructed at server start time, or an array of length zero if there are no start classes.

Example

The following example demonstrates how to use the *getStartClassInstances* method:

```
Object objs[] = sc.getStartClassInstances();
int i;
for (i=0; i < objs.length; i += 1) {
   if (objs[i] instanceof MyClass) {
       // Use class.
   }
}
```

1.9.9 makeConnection() Method

Opens and returns a new server connection.

Syntax

```
public java.sql.Connection makeConnection () throws SQLException
```

Returns

The newly created server connection.

Exceptions

java.sql.SQLException Thrown if an error occurred while opening a new connection.

Remarks

This connection is owned by the user java code. It must be committed and closed by the user.

To access the server context use the *DBConnectionContext.getServerContext* method on the *DBConnectionContext* for the current connection.

Note

Opening a connection can be expensive. Write your logic so that the number of calls to this method are minimized.

1.9.10 removeErrorListener(LogListener) Method

Removes the specified LogListener object from the list of Listener objects that receive notifications when an error is printed.

Syntax

```
public void removeErrorListener (LogListener ll)
```

Parameters

ll The Listener object to stop notifying.

Example

The following code removes a *LogListener* object from the list of error Listener objects.

```
ServerContext serverContext;
LogListener myErrorListener
serverContext.removeErrorListener(myErrorListener);
```

1.9.11 removeInfoListener(LogListener) Method

Removes the specified LogListener object from the list of Listener objects that receive notifications when info is printed.

Syntax

```
public void removeInfoListener (LogListener ll)
```

Parameters

ll The Listener object to stop notifying.

Example

The following code removes a *LogListener* object from the list of info Listener objects:

```
ServerContext serverContext;
LogListener myInfoListener
serverContext.removeInfoListener(myInfoListener);
```

1.9.12 removeShutdownListener(ShutdownListener) Method

Removes the specified ShutdownListener object from the list of Listener objects that receive notifications before this ServerContext is destroyed.

Syntax

```
public void removeShutdownListener (ShutdownListener sl)
```

Parameters

sl The Listener object to stop notifying.

Example

The following code removes a *ShutdownListener* object from the list of Listener objects that are to receive notification before the *ServerContext* is destroyed:

```
ServerContext serverContext;
ShutdownListener myShutdownListener
serverContext.removeShutdownListener(myShutdownListener);
```

1.9.13 removeWarningListener(LogListener) Method

Removes the specified LogListener object from the list of Listener objects that receive notifications when a warning is printed.

Syntax

```
public void removeWarningListener (LogListener ll)
```

Parameters

ll The Listener object to stop notifying.

Example

The following code removes a *LogListener* object from the list of warning Listener objects.

```
ServerContext serverContext;
LogListener myWarningListener
serverContext.removeWarningListener(myWarningListener);
```

1.9.14 shutdown() Method

Forces the server to shut down.

Syntax

```
public void shutdown ()
```

Remarks

Any registered *ShutdownListener* object have their *shutdownPerformed* method called.

Example

The following code forces the server to shut down:

```
ServerContext serverContext;
serverContext.shutdown();
```

Related Information

shutdownPerformed(ServerContext) Method [page 56]

1.10 ServerException Class

Thrown to indicate that there is an error condition that makes any further synchronization on the server impossible.

Package

```
com.sap.ml.script
```

Syntax

```
public class ServerException
```

Members

All members of ServerException, including inherited members.

Constructors

Modifier and Type	Constructor	Description
public	ServerException [page 54]	Constructs a ServerException.

Remarks

Throwing this exception causes the MobiLink server to shut down.

Example

The following example demonstrates how to throw a *ServerException* when a fatal problem occurs, and shut down the MobiLink server:

```
public void handleUpload(UploadData ud)
   throws SQLException, IOException, ServerException
{
UploadedTableData tables[] = ud.getUploadedTables();
if (tables == null) {
   throw new ServerException("Failed to read uploaded tables");
}
for (int i = 0; i < tables.length; i++) {
   UploadedTableData currentTable = tables[i];
```

```
    println("table " + java.lang.Integer.toString(i)
        + " name: " + currentTable.getName());
    // Print out delete result set.
    println("Deletes");
    printRSInfo(currentTable.getDeletes());
    // Print out insert result set.
    println("Inserts");
    printRSInfo(currentTable.getInserts());
    // print out update result set
    println("Updates");
    printUpdateRSInfo(currentTable.getUpdates());
}
}
```

In this section:

1.10.1 ServerException Constructor

Constructs a ServerException.

Overload list

Modifier and Type	Overload name	Description
public	ServerException() [page 54]	Constructs a ServerException with no detailed message.
public	ServerException(String) [page 55]	Constructs a ServerException with the specified detailed message.

In this section:

1.10.1.1 ServerException() Constructor

Constructs a ServerException with no detailed message.

Syntax

```
public ServerException ()
```

1.10.1.2 ServerException(String) Constructor

Constructs a ServerException with the specified detailed message.

Syntax

```
public ServerException (String s)
```

Parameters

s A detailed message.

1.11 ShutdownListener Interface

The Listener interface for catching server shutdowns.

Package

```
com.sap.ml.script
```

Syntax

```
public interface ShutdownListener
```

Members

All members of ShutdownListener, including inherited members.

Methods

Modifier and Type	Method	Description
public void	shutdownPerformed(ServerContext) [page 56]	Invoked before the ServerContext is destroyed due to server shutdown.

Remarks

Use this interface to ensure that all threads, connections, and other resources are cleaned up before the synchronization server exits.

Example

The following example demonstrates how to install a *ShutdownListener* object for the *ServerContext* instance:

```
class MyShutdownListener implements ShutdownListener {
   FileOutputStream _outFile;
   public MyShutdownListener(FileOutputStream outFile) {
       _outFile = outFile;
   }
   public void shutdownPerformed(ServerContext sc) {
       // Add shutdown code
       try {
           _outFile.write(("Shutting Down" + "\n").getBytes());
           _outFile.flush();
       }
       catch(Exception e) {
           // Print some error output to the synchronization server log.
           e.printStackTrace();
       }
       // ...
   }
}
```

The following code registers a *MyShutdownListener* object. Call this code from anywhere that has access to the *ServerContext* such as a class constructor or synchronization script.

```
ServerContext serv_context;
FileOutputStream outFile;
serv_context.addShutdownListener(new MyShutdownListener(outFile));
```

In this section:

1.11.1 shutdownPerformed(ServerContext) Method

Invoked before the ServerContext is destroyed due to server shutdown.

Syntax

```
public void shutdownPerformed (ServerContext sc)
```

Parameters

sc The context for the server that is being shut down.

1.12 SpatialUtilities Class

A collection of static methods to work with spatial values.

Package

```
com.sap.ml.script
```

Syntax

```
public class SpatialUtilities
```

Members

All members of SpatialUtilities, including inherited members.

Methods

Modifier and Type	Method	Description
public static byte[]	createSpatialValue(int, byte[]) [page 58]	Returns a new byte array that contains a spatial value formatted for download.
public static byte[]	getBytes(byte[]) [page 58]	Returns a new byte array with the same spatial data as the given byte array, but with the SRID removed.
public static int	getSRID(byte[]) [page 59]	Returns the SRID for the given spatial value.
public static void	setSRID(byte[], int) [page 59]	Stores the given SRID in the first four bytes of the given byte array.

In this section:

createSpatialValue(int, byte[]) Method [page 58]
Returns a new byte array that contains a spatial value formatted for download.

getBytes(byte[]) Method [page 58]
Returns a new byte array with the same spatial data as the given byte array, but with the SRID removed.

getSRID(byte[]) Method [page 59]
Returns the SRID for the given spatial value.

setSRID(byte[], int) Method [page 59]
Stores the given SRID in the first four bytes of the given byte array.

1.12.1 createSpatialValue(int, byte[]) Method

Returns a new byte array that contains a spatial value formatted for download.

Syntax

```
public static byte[] createSpatialValue (
    int srid,
    byte[] spatial_value
)
```

Parameters

srid The srid.

spatial_value The spatial data.

Returns

The spatial value formatted for download.

Remarks

The first four bytes contain the given SRID in little endian, and the remainder is the spatial data passed in the given byte array.

1.12.2 getBytes(byte[]) Method

Returns a new byte array with the same spatial data as the given byte array, but with the SRID removed.

Syntax

```
public static byte[] getBytes (byte[] spatial_value)
```

Parameters

spatial_value A spatial value that needs its SRID removed.

Returns

The new byte array.

1.12.3 getSRID(byte[]) Method

Returns the SRID for the given spatial value.

Syntax

```
public static int getSRID (byte[] spatial_value)
```

Parameters

spatial_value The uploaded value. The first four bytes must contain the SRID encoded in little endian.

Returns

The SRID.

1.12.4 setSRID(byte[], int) Method

Stores the given SRID in the first four bytes of the given byte array.

Syntax

```
public static void setSRID (
    byte[] spatial_value,
    int srid
)
```

Parameters

spatial_value The array to store the SRID in.
srid The SRID to store.

1.13 TimestampWithTimeZone Class

A *java.sql.Timestamp* with methods to get and set the time zone.

Package

```
com.sap.ml.script
```

Syntax

```
public class TimestampWithTimeZone
```

Members

All members of TimestampWithTimeZone, including inherited members.

Variables

Modifier and Type	Variable	Description
protected static Pattern	_timestamp_pattern	
protected static Pattern	_timezone_pattern	
protected static Pattern	_time_pattern	
protected static Pattern	_date_pattern	

Constructors

Modifier and Type	Constructor	Description
public	TimestampWithTimeZone(int, int, int, int, int, int, int, int, int) [page 62]	Constructs a new TimestampWithTimeZone with the specified year, month, date, hour, minute, second, nano, time zone hour and time zone minute.

Methods

Modifier and Type	Method	Description
public boolean	equals [page 63]	Returns true if o is equal to the Timestamp portion of this and either o is a TimestampWithTimeZone with the same time zone offset as this, or o is not a TimestampWithTimeZone and this has a time zone offset of 00:00.
public int	getTimeZoneOffsetHours() [page 65]	Gets the hours portion of the time zone offset.
public int	getTimeZoneOffsetMinutes() [page 65]	Gets the minutes portion of the time zone offset.
public void	setTimeZoneOffsetHours(int) [page 66]	Sets the hours portion of the time zone offset.
public void	setTimeZoneOffsetMinutes(int) [page 66]	Sets the minutes portion of the time zone offset.
public String	toString() [page 67]	Returns the string representing this timestamp with the form yyyy-mm-dd hh:mm:ss.ffffff Shh:mm, where S is the sign of the hours field.
public static TimestampWithTimeZone	toTimestampWithTimeZone(Timestamp) [page 67]	Converts the given Timestamp to a TimestampWithTimeZone.
public static TimestampWithTimeZone	valueOf(String) [page 67]	Converts a String to a TimestampWithTimeZone value.

Remarks

Use this when using the direct row API to specify the time zone offset for TIMESTAMP WITH TIME ZONE columns. *PreparedStatement* and *ResultSet* objects from JDBC drivers other than the direct row API treat this as a normal *Timestamp*.

In this section:

TimestampWithTimeZone(int, int, int, int, int, int, int, int, int) Constructor [page 62]
Constructs a new TimestampWithTimeZone with the specified year, month, date, hour, minute, second, nano, time zone hour and time zone minute.

equals Method [page 63]
Returns true if o is equal to the Timestamp portion of this and either o is a TimestampWithTimeZone with the same time zone offset as this, or o is not a TimestampWithTimeZone and this has a time zone offset of 00:00.

getTimeZoneOffsetHours() Method [page 65]
Gets the hours portion of the time zone offset.

getTimeZoneOffsetMinutes() Method [page 65]
Gets the minutes portion of the time zone offset.

setTimeZoneOffsetHours(int) Method [page 66]

Sets the hours portion of the time zone offset.

setTimeZoneOffsetMinutes(int) Method [page 66]
Sets the minutes portion of the time zone offset.

toString() Method [page 67]
Returns the string representing this timestamp with the form yyyy-mm-dd hh:mm:ss.ffffff Shh:mm, where S is the sign of the hours field.

toTimestampWithTimeZone(Timestamp) Method [page 67]
Converts the given Timestamp to a TimestampWithTimeZone.

valueOf(String) Method [page 67]
Converts a String to a TimestampWithTimeZone value.

1.13.1 TimestampWithTimeZone(int, int, int, int, int, int, int, int, int) Constructor

Constructs a new TimestampWithTimeZone with the specified year, month, date, hour, minute, second, nano, time zone hour and time zone minute.

Syntax

```
public TimestampWithTimeZone (
    int year,
    int month,
    int date,
    int hour,
    int minute,
    int second,
    int nano,
    int tz_hour,
    int tz_minute
)
```

Parameters

year The year minus 1900.
month An integer ranged from 0 to 11.
date An integer ranged from 1 to 31.
hour An integer ranged from 0 to 23.
minute An integer ranged from 0 to 59.
second An integer ranged from 0 to 59.
nano An integer ranged from 0 to 999,999,999.
tz_hour An integer ranged from -14 to 14.
tz_minute An integer ranged from 0 to 59.

Exceptions

java.lang.IllegalArgumentException Thrown if *tz_minute* is not in 0 - 59 or *tz_hour* is not within the appropriate range.

1.13.2 equals Method

Returns true if o is equal to the Timestamp portion of this and either o is a TimestampWithTimeZone with the same time zone offset as this, or o is not a TimestampWithTimeZone and this has a time zone offset of 00:00.

Overload list

Modifier and Type	Overload name	Description
public boolean	equals(Object) [page 64]	Returns true if o is equal to the Timestamp portion of this and either o is a TimestampWithTimeZone with the same time zone offset as this, or o is not a TimestampWithTimeZone and this has a time zone offset of 00:00.
public boolean	equals(Timestamp) [page 64]	Returns true if o is equal to the Timestamp portion of this and either o is a TimestampWithTimeZone with the same time zone offset as this, or o is not a TimestampWithTimeZone and this has a time zone offset of 00:00.

In this section:

equals(Object) Method [page 64]
Returns true if o is equal to the Timestamp portion of this and either o is a TimestampWithTimeZone with the same time zone offset as this, or o is not a TimestampWithTimeZone and this has a time zone offset of 00:00.

equals(Timestamp) Method [page 64]
Returns true if o is equal to the Timestamp portion of this and either o is a TimestampWithTimeZone with the same time zone offset as this, or o is not a TimestampWithTimeZone and this has a time zone offset of 00:00.

1.13.2.1 equals(Object) Method

Returns true if o is equal to the Timestamp portion of this and either o is a TimestampWithTimeZone with the same time zone offset as this, or o is not a TimestampWithTimeZone and this has a time zone offset of 00:00.

Syntax

```
public boolean equals (Object o)
```

Parameters

o The object to compare against.

Returns

True if o is equal to this; otherwise false.

Related Information

equals(Timestamp) Method [page 64]

1.13.2.2 equals(Timestamp) Method

Returns true if o is equal to the Timestamp portion of this and either o is a TimestampWithTimeZone with the same time zone offset as this, or o is not a TimestampWithTimeZone and this has a time zone offset of 00:00.

Syntax

```
public boolean equals (Timestamp o)
```

Parameters

o The object to compare against.

Returns

True if o is equal to this; otherwise false.

Related Information

equals(Object) Method [page 64]

1.13.3 getTimeZoneOffsetHours() Method

Gets the hours portion of the time zone offset.

Syntax

```
public int getTimeZoneOffsetHours ()
```

Returns

The hours portion of the time zone offset.

1.13.4 getTimeZoneOffsetMinutes() Method

Gets the minutes portion of the time zone offset.

Syntax

```
public int getTimeZoneOffsetMinutes ()
```

Returns

The minutes portion of the time zone offset.

1.13.5 setTimeZoneOffsetHours(int) Method

Sets the hours portion of the time zone offset.

Syntax

```
public void setTimeZoneOffsetHours (int tz_hour)
```

Parameters

tz_hour The new hours portion of the time zone offset.

Exceptions

java.lang.IllegalArgumentException Thrown if *tz_hour* is not in -14 - 14.

1.13.6 setTimeZoneOffsetMinutes(int) Method

Sets the minutes portion of the time zone offset.

Syntax

```
public void setTimeZoneOffsetMinutes (int tz_minute)
```

Parameters

tz_minute The new minutes portion of the time zone offset.

Exceptions

java.lang.IllegalArgumentException Thrown if *tz_minute* is not in 0 - 59.

1.13.7 toString() Method

Returns the string representing this timestamp with the form yyyy-mm-dd hh:mm:ss.ffffff Shh:mm, where S is the sign of the hours field.

Syntax

```
public String toString ()
```

Returns

The string representing this timestamp.

1.13.8 toTimestampWithTimeZone(Timestamp) Method

Converts the given Timestamp to a TimestampWithTimeZone.

Syntax

```
public static TimestampWithTimeZone toTimestampWithTimeZone (Timestamp ts)
```

Parameters

ts The timestamp to convert.

Returns

ts if *ts* is an instance of *TimestampWithTimeZone*. Otherwise, constructs a new *TimestampWithTimeZone* that is equivalent to *ts* with a time zone offset of 00:00.

1.13.9 valueOf(String) Method

Converts a String to a TimestampWithTimeZone value.

Syntax

```
public static TimestampWithTimeZone valueOf (String val)
```

Parameters

val A timestamp with time zone in the *yyyy-mm-dd hh:mm:ss[.f...][[-+]hh:mm]* format. The fractional and time zone parts may be omitted. If the time zone is present, the sign can be omitted.

Returns

The new TimestampWithTimeZone.

Exceptions

java.lang.IllegalArgumentException Thrown if *val* does not have the correct format.

1.14 UpdateResultSet Interface

A result set object that includes special methods for accessing the pre-image (old) and post-image (new) values of a specified row.

Package

```
com.sap.ml.script
```

Syntax

```
public interface UpdateResultSet extends
```

Members

All members of UpdateResultSet, including inherited members.

Methods

Modifier and Type	Method	Description
public void	setNewRowValues() [page 69]	Sets the mode of this result set to return new column values(the post update row).
public void	setOldRowValues() [page 70]	Sets the mode of this result set to return old column values (the pre update row).

Remarks

You can hold the update operations for one upload transaction for one table.

New and old rows can both be accessed by changing the mode of the *ResultSet* to old or new.

Use the *DownloadTableData.getUpdates* method to obtain an *UpdateResultSet* instance.

UpdateResultSet extends *java.sql.ResultSet* and adds the *setNewRowValues* and *setOldRowValues* methods. Otherwise it can be used as a regular *ResultSet*.

Consult your Java Software Development Kit documentation for more information about *java.sql.ResultSet*.

In this section:

setNewRowValues() Method [page 69]
: Sets the mode of this result set to return new column values(the post update row).

setOldRowValues() Method [page 70]
: Sets the mode of this result set to return old column values (the pre update row).

Related Information

getUpdates() Method [page 80]

1.14.1 setNewRowValues() Method

Sets the mode of this result set to return new column values(the post update row).

Syntax

```
public void setNewRowValues ()
```

Remarks

The result set represents the latest updated values in the remote client database.

This is the default mode.

Example

The following code shows how to set the mode of the *UpdateResultSet* to return new column values:

```
UpdateResultSet results
results.setNewRowValues();
```

1.14.2 setOldRowValues() Method

Sets the mode of this result set to return old column values (the pre update row).

Syntax

```
public void setOldRowValues ()
```

Remarks

In this mode, the *UpdateResultSet* represents old column values obtained by the client in the last synchronization.

Example

The following code shows how to set the mode of the *UpdateResultSet* to return old column values:

```
UpdateResultSet results
results.setOldRowValues();
```

1.15 UploadData Interface

Encapsulates upload operations for direct row handling.

Package

```
com.sap.ml.script
```

Syntax

```
public interface UploadData
```

Members

All members of UploadData, including inherited members.

Methods

Modifier and Type	Method	Description
public UploadedTableData	getUploadedTableByName(String) [page 72]	Returns an UploadedTableData instance representing the specified table.
public UploadedTableData[]	getUploadedTables() [page 73]	Returns an array of UploadedTableData objects for the current synchronization.

Remarks

An *UploadData* instance representing a single upload transaction is passed to the handle_UploadData event.

Note

You must handle direct row handling upload operations in the method registered for the handle_UploadData event. The *UploadData* is destroyed after each call to the registered method. Do not create a new instance of *UploadData* to use in subsequent events.

Use the *getUploadedTables* or *getUploadedTableByName* methods to obtain *UploadedTableData* instances.

A synchronization has one *UploadData* unless the remote database is using transactional upload.

In this section:

Returns an UploadedTableData instance representing the specified table.

getUploadedTables() Method [page 73]
Returns an array of UploadedTableData objects for the current synchronization.

Related Information

UploadedTableData Interface [page 74]

1.15.1 getUploadedTableByName(String) Method

Returns an UploadedTableData instance representing the specified table.

Syntax

```
public UploadedTableData getUploadedTableByName (String table_name)
```

Parameters

table_name The name of the uploaded table for which you want the uploaded data.

Returns

An *UploadedTableData* instance representing the specified table, or null if a table of the given name does not exist for the current synchronization.

Example

Assume you use a method named *HandleUpload* for the handle_UploadData event. The following example uses the *GetUploadedTableByName* method to return an *UploadedTableData* instance for the *remoteOrderstable*.

```
public void handleUpload(UploadData ud)
   throws SQLException, IOException
{
   UploadedTableData uploaded_t1 = ud.GetUploadedTableByName("remoteOrders");
   //...
}
```

Related Information

UploadedTableData Interface [page 74]

1.15.2 getUploadedTables() Method

Returns an array of UploadedTableData objects for the current synchronization.

Syntax

```
public UploadedTableData[] getUploadedTables ()
```

Returns

An array of *UploadedTableData* objects for the current synchronization. The order of tables in the array is the same as the upload order of the client.

Remarks

The order to the tables in the array is the same order that the synchronization server uses for SQL row handling, and is the optimal order for preventing referential integrity violations. If your data source is a relational database, use this table order.

Example

Assume you use a method named *HandleUpload* for the handle_UploadData event. The following example uses the *getUploadedTables* method to return *UploadedTableData* instances for the current upload transaction.

```
public void handleUpload(UploadData ud)
   throws SQLException, IOException
{
   UploadedTableData tables[] = ud.getUploadedTables();
   //...
}
```

Related Information

UploadedTableData Interface [page 74]

1.16 UploadedTableData Interface

Encapsulates table operations for direct row handling uploads.

Package

```
com.sap.ml.script
```

Syntax

```
public interface UploadedTableData
```

Members

All members of UploadedTableData, including inherited members.

Methods

Modifier and Type	Method	Description
public java.sql.ResultSet	getDeletes() [page 76]	Returns a java.sql.ResultSet object representing delete operations uploaded by a synchronization client.
public java.sql.ResultSet	getInserts() [page 77]	Returns a java.sql.ResultSet object representing insert operations uploaded by a synchronization client.
public java.sql.ResultSetMetaData	getMetaData() [page 78]	Gets the metadata for the UploadedTableData instance.
public String	getName() [page 79]	Returns the table name for the UploadedTableData instance.
public UpdateResultSet	getUpdates() [page 80]	Returns a UpdateResultSet object representing update operations uploaded by a synchronization client.

Remarks

You can use an *UploadedTableData* instance to obtain a table's insert, update, and delete operations for a single upload transaction. Use the *getInserts*, *getUpdates*, and *getDeletes* methods to return standard JDBC *java.sql.ResultSet* objects.

Consult your Java Software Development Kit documentation for more information about *java.sql.ResultSet* and *java.sql.ResultSetMetaData*.

Table metadata can be accessed using the *getMetaData* method or the result sets returned by *getInserts*, *getUpdates*, and *getDeletes*. The delete result set only includes primary key columns for a table.

Example

The following code gets the deletes uploaded and prints out the first column of each:

```
void printFirstColOfDeletes( UploadedTableData tab_data )
{
   ResultSet deletes = tab_data.getDeletes();
   while( deletes.next() ) {
       java.lang.System.out.println( deletes.getString( 1 ) );
   }
   deletes.close();
}
```

The following code prints the new and old value of the third column of each update:

```
void printThirdColOfUpdates( UploadedTableData tab_data )
{
   ResultSet updates = tab_data.getUpdates();
   while( updates.next() ) {
       updates.setOldRowValues();
       java.lang.System.out.println( "old row col: " + updates.getString( 3 ) );
       updates.setNewRowValues();
       java.lang.System.out.println( "new row col: " + updates.getString( 3 ) );
   }
   updates.close();
}
```

In this section:

Related Information

UploadData Interface [page 71]

1.16.1 getDeletes() Method

Returns a java.sql.ResultSet object representing delete operations uploaded by a synchronization client.

Syntax

```
public java.sql.ResultSet getDeletes ()
```

Returns

A java.sql.ResultSet object that represents delete operations uploaded by a synchronization client.

Remarks

The result set contains primary key values for deleted rows.

Example

Assume your remote client contains a table named *remoteOrders*. The following example uses the *DownloadTableData.getDeletes* method to obtain a result set of deleted rows. In this case, the delete result set includes a single primary key column.

```
import ianywhere.ml.script.*;
import java.sql.*;
// The method used for the handle_UploadData event.
public void HandleUpload(UploadData ut)
   throws SQLException, IOException
{
   // Get an UploadedTableData for the remoteOrders table.
   UploadedTableData remoteOrdersTable =
ut.getUploadedTableByName("remoteOrders");
   // Get deletes uploaded by the synchronization client.
   java.sql.ResultSet delete_rs = remoteOrdersTable.getDeletes();
    while (delete_rs.next()) {
        // Get primary key values for deleted rows.
        int deleted_id = delete_rs.getInt(1);
        // ...
   }
   delete_rs.close();
}
```

1.16.2 getInserts() Method

Returns a java.sql.ResultSet object representing insert operations uploaded by a synchronization client.

Syntax

```
public java.sql.ResultSet getInserts ()
```

Returns

A *java.sql.ResultSet* object representing insert operations uploaded by a synchronization client.

Remarks

Each Insert is represented by one row in the result set.

Example

Assume your remote client contains a table named *remoteOrders*. The following example uses the *UploadedTableData.getInserts* method to obtain a result set of inserted rows. The code obtains the order amount for each row in the current upload transaction.

```
import com.sap.ml.script.*;
import java.sql.*;
// The method used for the handle_UploadData event
public void HandleUpload(UploadData ut)
   throws SQLException, IOException
{
   // Get an UploadedTableData instance representing the remoteOrders table.
   UploadedTableData remoteOrdersTable =
ut.getUploadedTableByName("remoteOrders");
   // Get inserts uploaded by the synchronization client.
   java.sql.ResultSet rs = remoteOrdersTable.getInserts();
   while (rs.next()) {
       // get the uploaded order_amount
       double order_amount = rs.getDouble("order_amount");
       // ...
   }
   rs.close();
}
```

1.16.3 getMetaData() Method

Gets the metadata for the UploadedTableData instance.

Syntax

```
public java.sql.ResultSetMetaData getMetaData ()
```

Returns

The metadata for the *UploadedTableData* instance.

Remarks

The metadata is a standard *java.sql.ResultSetMetaData* instance.

If you want the *ResultSetMetaData* to contain column name information, you must specify the client option to send column names.

Consult your Java Software Development Kit documentation for more information about *java.sql.ResultSetMetaData*.

Example

The following example obtains a *java.sql.ResultSetMetaData* instance for an uploaded table named *remoteOrders*. The code uses the *ResultSetMetaData.getColumnCount* and *getColumnLabel* methods to compile a list of column names.

```
import ianywhere.ml.script.*;
import java.sql.*;
// The method used for the handle_UploadData event.
public void HandleUpload(UploadData ut) {
   throws SQLException, IOException
{
   // Get an UploadedTableData instance representing the remoteOrders table.
   UploadedTableData remoteOrdersTable =
ut.getUploadedTableByName("remoteOrders");
   // get inserts uploaded by the synchronization client
   java.sql.ResultSet rs = remoteOrdersTable.getInserts();
   // Obtain the result set metadata.
   java.sql.ResultSetMetaData md = rs.getMetaData();
   String columnHeading = "";
   // Compile a list of column names.
   for (int c=1; c <= md.getColumnCount(); c += 1) {
       columnHeading += md.getColumnLabel( c );
       if (c < md.getColumnCount()) {
           columnHeading += ", ";
       }
```

```
    }
    //...
}
```

In this case, a method named *HandleUpload* handles the handle_UploadData synchronization event.

1.16.4 getName() Method

Returns the table name for the UploadedTableData instance.

Syntax

```
public String getName ()
```

Returns

The table name for the *UploadedTableData* instance.

Remarks

You can also access the table name using the *java.sql.ResultSetMetaData* instance returned by the *getMetaData* method.

Example

The following example obtains the name of each uploaded table in a single upload transaction:

```
import ianywhere.ml.script.*;
import java.sql.*;
// The method used for the handle_UploadData event.
public void HandleUpload(UploadData ud) {
   throws SQLException, IOException
{
   int i;
   // Get UploadedTableData instances.
   UploadedTableData tables[] = ud.getUploadedTables();
   for (i=0; i<tables.length; i+=1) {
       // Get the table name.
       String table_name = tables[i].getName();
       // ...
   }
}
```

Related Information

getMetaData() Method [page 78]

1.16.5 getUpdates() Method

Returns a UpdateResultSet object representing update operations uploaded by a synchronization client.

Syntax

```
public UpdateResultSet getUpdates ()
```

Returns

An *UpdateResultSet* object representing update operations uploaded by a synchronization client.

Remarks

Each update is represented by one row including all column values. *UpdateResultSet* extends *java.sql.ResultSet* to include special methods for synchronization conflict detection.

Example

Assume your remote client contains a table named *remoteOrders*. The following example uses the *getUpdates* method to obtain a result set of updated rows. The code obtains the order amount for each row.

```
import ianywhere.ml.script.*;
import java.sql.*;
// The method used for the handle_UploadData event.
public void HandleUpload(UploadData ut)
   throws SQLException, IOException
{
   // Get an UploadedTableData instance representing the remoteOrders table.
   UploadedTableData remoteOrdersTable =
ut.getUploadedTableByName("remoteOrders");
   // Get inserts uploaded by the synchronization client.
   java.sql.ResultSet rs = remoteOrdersTable.getUpdates();
   while (rs.next()) {
       // Get the uploaded order_amount.
       double order_amount = rs.getDouble("order_amount");
       // ...
   }
   rs.close();
}
```

Related Information

UpdateResultSet Interface [page 68]

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon  : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon  : You are leaving the documentation for that particular SAP product or service and are entering a SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

www.sap.com/contactsap

THE BEST RUN SAP